VŠB – Technická univerzita Ostrava
Fakulta strojní
Katedra automatizační techniky a řízení

Vývoj mobilní aplikace pro OS Android

Model Control Using Mobile Application

Student: Bc. Jan Nádvorník
Vedoucí diplomové práce: Ing. Pavel Smutný, Ph.D.

Ostrava 2013

VŠB - Technická univerzita Ostrava
Fakulta strojní
Katedra automatizační techniky a řízení

Zadání diplomové práce

Student: **Bc. Jan Nádvorník**

Studijní program: N2301 Strojní inženýrství

Studijní obor: 3902T004 Automatické řízení a inženýrská informatika

Téma: Ovládání modelu pomocí mobilní aplikace
Model Control Using Mobile Application

Zásady pro vypracování:

1. Popište problematiku vývoje mobilních aplikací.
2. Charakterizujte vývojové prostředí a konstrukci modelů různých výrobců.
3. Popište druhy komunikace mezi modelem a mobilním zařízením.
4. Navrhněte a popište úlohu, kterou zpracujete v podobě mobilní aplikace.
5. Zhodnoťte dosažené výsledky a navrhněte směry dalšího řešení.

Seznam doporučené odborné literatury:

Android.com. Developers Guide. Mountain View, California, USA Dostupné z: http://developer.android.com/guide/index.html

Murphy, Mark, L. Android 2: Průvodce programováním mobilních aplikací. Brno: Computer Press, 2011. ISBN 978-80-251-3194-7. Dostupné z: http://knihy.cpress.cz/android-2.html.

Ujbányai, M. Programujeme pro Android. Praha: Grada Publishing, 2012. ISBN 978-80-247-3995-3

Rusek, J. Komunikace bezdrátových snímačů s podporou OS Android [online]. 2012 [cit. 2012-30-10]. Diplomová práce. VŠB-TU Ostrava, Vedoucí práce Jiří David. Dostupné z: http://dspace.vsb.cz/handle/10084/93591

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Pavel Smutný, Ph.D.**

Datum zadání: 14.12.2012
Datum odevzdání: 20.05.2013

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta strojní 3 OSTRAVA

Tůma

prof. Ing. Jiří Tůma, CSc.
vedoucí katedry

doc. Ing. Ivo Hlavatý, Ph.D.
děkan fakulty

Místopřísežné prohlášení studenta

Prohlašuji, že jsem celou diplomovou práci včetně příloh vypracoval samostatně pod vedením vedoucího diplomové práce a uvedl jsem všechny použité podklady a literaturu.

V Ostravě: 17. května 2013 ……

podpis studenta

Prohlašuji, že

- jsem byl seznámen s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., autorský zákon, zejména § 35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a § 60 – školní dílo.
- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen „VŠB-TUO“) má právo nevýdělečně ke své vnitřní potřebě diplomovou práci užít (§ 35 odst. 3).
- souhlasím s tím, že diplomová práce bude v elektronické podobě uložena v Ústřední knihovně VŠB-TUO k nahlédnutí a jeden výtisk bude uložen u vedoucího diplomové práce. Souhlasím s tím, že údaje o kvalifikační práci budou zveřejněny v informačním systému VŠB-TUO.
- bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona.
- bylo sjednáno, že užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).
- beru na vědomí, že odevzdáním své práce souhlasím se zveřejněním své práce podle zákona č.111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, bez ohledu na výsledek její obhajoby.

V Ostravě: 17. května 2013

……

podpis

Jan Nádvorník
Zahradní 613
798 52 Konice

Poděkování

Děkuji vedoucímu Diplomové práce panu Ing. Pavlu Smutnému, Ph.D. za odborné vedení této práce a za věcné rady, které mě vždy posunuly o krok dále. Rád bych také poděkoval své rodině, která mi vytvářela výborné podmínky nejen při psaní této práce, ale také po celou dobu studia na VŠB – TUO.

ANOTACE DIPLOMOVÉ PRÁCE

NÁDVORNÍK, J. *Ovládání modelu pomocí mobilní aplikace: diplomová práce.* Ostrava: VŠB – Technická univerzita Ostrava, Fakulta strojní, Katedra automatizační techniky a řízení, 2013, 45 s. Vedoucí práce: Smutný P.

Hlavní náplní této práce byl návrh a realizace mobilní aplikace pro operační systém Android, která je zaměřena na manuální ovládání mobilního robotu pomocí bezdrátového rozhraní Bluetooth. Aplikace umožňuje ovládání robotu interakcí s displejem, nebo hlasem. Pomocí grafického rozhraní aplikace je možné sledovat aktuální vzdálenost robotu od překážky. Měření vzdálenosti je realizováno ultrazvukovým senzorem umístěným v přední části robotu. Pro vývoj aplikace bylo zapotřebí sestavit prototyp mobilního robotu. K tomuto účelu bylo využito stavebnice Lego Mindstorms. Prototyp mobilního robotu je založen na diferenciálním podvozku.

ANNOTATION OF MASTER THESIS

NÁDVORNÍK, J. *Model Control Using Mobile Application: Master Thesis.* Ostrava: VŠB – Technical University of Ostrava, Faculty of Mechanical Engineering, Department of Control Systems and Instrumentation, 2013, 45 p. Thesis head: Smutný P.

The main concern of this work was the design and realization of the mobile application for the Android operating system, which is focused on manual control of mobile robot using wireless Bluetooth technology. The application allows the robot to control interaction with the display, or voice. When you use a graphical interface, you can monitor the current distance of the robot from obstacles. The measurement of distance is carried out ultrasonic sensor placed in front of the robot. It was necessary to build a prototype of a mobile robot for the development of the application. For this purpose was used Lego Mindstorms. The prototype of the mobile robot is based on the differential gear.

Obsah

SEZNAM POUŽITÝCH ZNAČEK A SYMBOLŮ

Ah	ampérhodina
APK	souborový formát (application package file)
API	Application Programming Interface
Bajt	osmiciferné binární číslo (byte)
bit	dvojková číslice (binary digit)
C	programovací jazyk
C++	objektově orientovaný programovací jazyk
cm	centimetr - jednotka délky
COBOL	programovací jazyk (common business oriented language)
flash	programovatelná paměť
Fortran	imperativní programovací jazyk **(**formula translator)
GPL	všeobecná veřejná licence (General Public License)
GPS	Global Positioning System
Hz	hertz - jednotka frekvence
IBM	International Business Machines Corporation
IT	informační technologie (information technology)
Java	objektově orientovaný programovací jazyk
kbps	jednotka přenosové rychlosti (bit za sekundu)
LED	dioda emitující světlo (Light-Emitting Diode)
MB	megabajt
ms	milisekunda
m/s	metr za sekundu
PC	osobní počítač (personal computer)
PHP	hypertextový procesor (Hypertext Preprocessor)
px	jednotka digitální rastrové grafiky (picture element)

Python	objektově orientovaný skriptovací programovací jazyk
RAM	paměť s přímým přístupem (random-access memory)
RCP	Rich Client Platform
RJ12	standardizovaný konektor
SDK	software development kit
SWT	knihovna grafických uživatelských prvků (Standard Widget Toolkit)
UNIX	ochranná známka operačního systému
USB	univerzální sériová sběrnice (Universal Serial Bus)
V	volt
Wi-Fi	technologie pro bezdrátovou komunikaci v počítačových sítích
XML	rozšiřitelný značkovací jazyk (Extensible Markup Language)

ÚVOD

Mobilní telefony prošly za posledních pár let znatelným vývojem a ze zařízení, které sloužily pouze pro telefonování, se staly kapesní počítače s vlastním operačním systémem. Od roku 2010 se na trhu výrazně prosazují také polohovací zařízení (tablety), které fungují na stejném operačním systému jako dnešní mobilní telefony, ale jsou rozměrově větší a funkčně odlišné. Kromě dotykového ovládání se do popředí dostává ovládání hlasem. Mobilní zařízení jsou prostřednictvím bezdrátových technologií Bluetooth a Wi-Fi schopny komunikovat s jakýmkoli zařízením, využívajícím tento standard. Této skutečnosti začali využívat výrobci různých stavebnicových systémů. Stavebnice byly rozšířeny o řídicí jednotky, servomotory a různé typy snímačů.

Tato práce se zabývá možnostmi vývoje mobilní aplikace pro operační systémy Android, iOS a Windows Mobile. Dále se věnuje konstrukci modelů vytvořených ze stavebnice Lego Mindstorms a popisu jednotlivých funkčních prvků. Hlavním úkolem této práce bylo vytvořit aplikaci pro operační systém Android. Následující kapitoly jsou tak věnovány vývojovému prostředí Eclipse, možnostem komunikace mezi modelem a mobilním zařízením a vývoji aplikace, pomocí které bude model ovládán.

1 VÝVOJ MOBILNÍCH APLIKACÍ

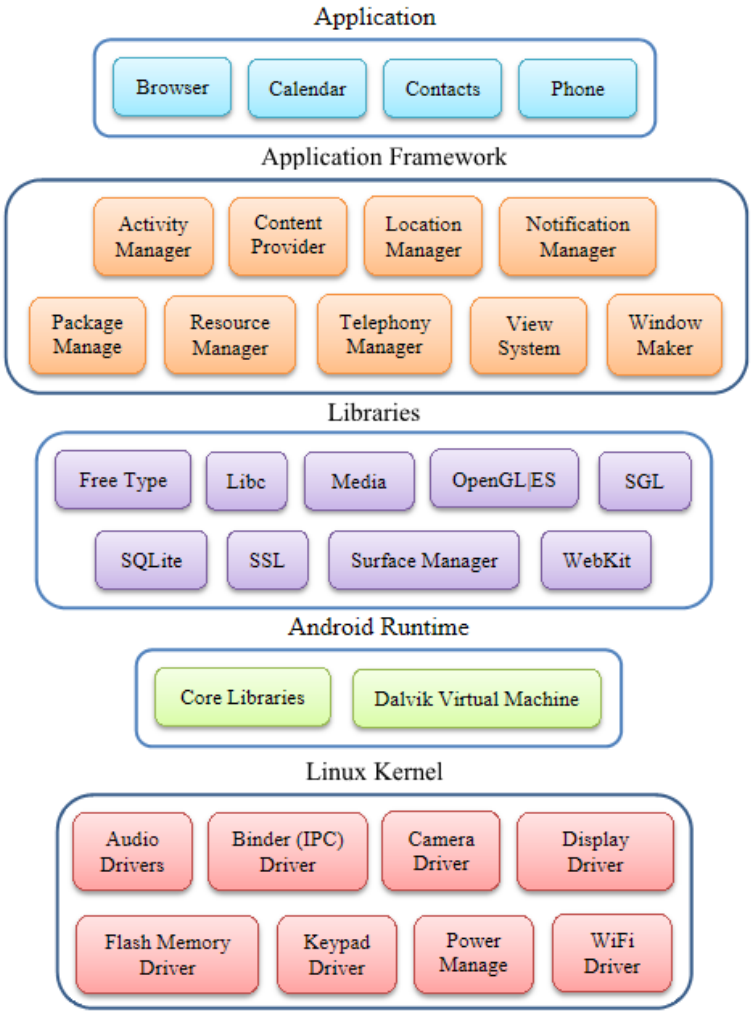
Prvopočátky chytrých telefonů se datují od roku 1993, kdy se na trhu objevil historicky první přístroj s vlastním operačním systémem. Jednalo se o komunikátor *IBM* Simon, (Obr. 1.1) který disponoval černobílým dotykovým displejem, který sloužil místo číselníku. Mezi základní funkce patřily aplikace jako kalendář, seznam kontaktů a kalkulačka. Od roku 2005 došlo k prudkému rozmachu mobilních operačních systémů. Mezi hlavní inovátory na trhu patří operační systémy Android, iOS a Windows Phone. [Kovařík, 2012]

Obr. 1.1 Osobní komunikátor IBM Simon [Opletal, 2012]

1.1 Operační systém Android

Společnost Android Inc. byla založena roku 2003. Jejím hlavním cílem bylo vytvoření mobilního zařízení, které klade důraz na lokaci, ve které se majitel nachází. Roku 2005 pak došlo k odkoupení společností Google. Následně vzniklo sdružení Open Handset Aliance, které sdružuje přední světové výrobce v oblasti *IT*. Na základě této spolupráce byla roku 2008 vydána první verze systému Android 1.0. O týden později byl uvolněn balíček *SDK* 1.0 určený pro vývojáře. Od té doby prošel systém mnoha změnami a inovacemi, rozšířil se na přenosné počítače a v prvním čtvrtletí roku 2013 bylo na trhu již 750 milionů aktivních zařízení s tímto operačním systémem.
[Bennett, 2012; Kovařík, 2012; Page, 2013]

Už od první dostupné verze je tento systém šířen pod licencí Apache a *GPL*. Zdrojové kódy systému jsou tak volně přístupné a může je používat a upravovat kdokoli. Ty jsou tak často využívány výrobci mobilních zařízení, nebo poskytovateli mobilních služeb, kteří mohou systém odladit podle svých představ. Architektura systému (Obr. 1.2) je rozdělena do několika vrstev. Linuxové jádro umožňuje běh více aplikací současně, přičemž každá je spuštěna jako samostatný proces. Android obsahuje také sadu *C*/*C++* knihoven, které umožňují přístup aplikací k různým komponentám systému. Další vrstvou je Android runtime, což je soubor základních knihoven a virtuální stroj Dalvik, který využívají aplikace pro svůj běh. Pro vývojáře nezbytný je aplikační Framework, který obsahuje objekty pro tvorbu logiky aplikace a uživatelského rozhraní. Kromě samotného operačního systému obsahuje Android aplikace jako kalendář, navigaci, atd. Systém si tak uživatel může rozšiřovat o další aplikace podle svých představ. [Tokár, 2011]



Obr. 1.2 Architektura systému Android

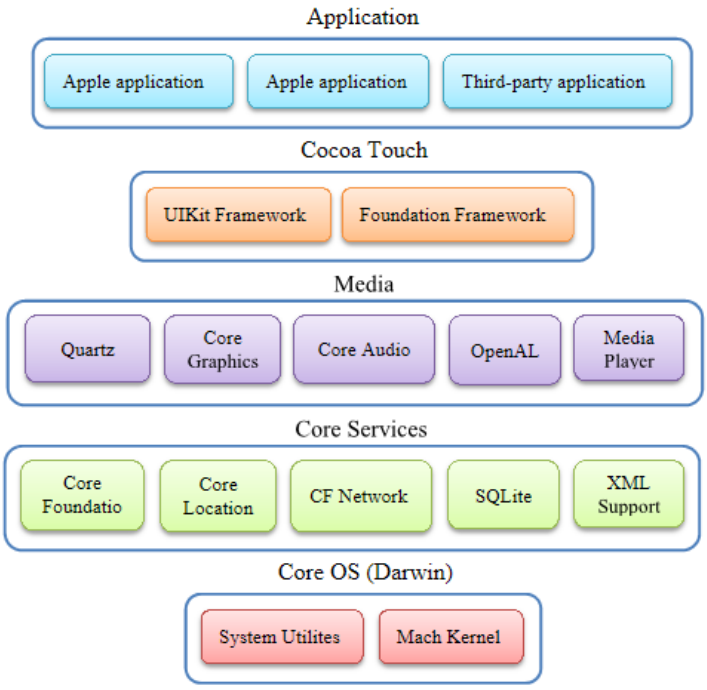
Pro vývoj aplikací pro tento operační systém je určený vývojářský balík *SDK*, který obsahuje sadu knihoven, emulátor zařízení, tutoriály a dokumentaci. *SDK* je volně ke stažení na internetových stránkách Android Developers. Doporučené prostředí pro vývoj

aplikací je Eclipse, pro který existuje řada zásuvných modulů. K programování je také možné využít jakýkoli textový editor a pro kompilaci a ladění aplikace příkazový řádek. Testování aplikací je možné pomocí emulátoru, nebo přímo na připojeném přístroji. Na rozdíl od iOS, nebo Windows Phone 8 je tato funkce pro testování aplikace zdarma. [Tokár, 2011]

1.2 Operační systém iOS

V roce 2007 vstoupila na trh společnost Apple se svým telefonem iPhone a mobilním operačním systémem iOS. Systém oslovil zákazníky především svou jednoduchostí, a rychlostí. [Kovařík, 2012]

Tento systém lze využívat pouze na zařízeních od společnosti Apple. Jedná se o odlehčenou verzi operačního systému určeného pro stolní počítače stejnojmenné společnosti. Systém je založený na *UNIX*ovém jádře Mach. Architektura operačního systému iOS (Obr. 1.3) se skládá z jádra operačního systému, které poskytuje nízkoúrovňové funkce. Další vrstvou v hierarchické struktuře je vrstva služeb poskytovaných jádrem. V této vrstvě je například podpora *XML* a využívají ji aplikace. Ve vrstvě médií jsou obsaženy technologie na tvorbu multimediálních aplikací. O základní definici struktury aplikace se stará vrstva Cocoa Touch. [Tokár, 2011]



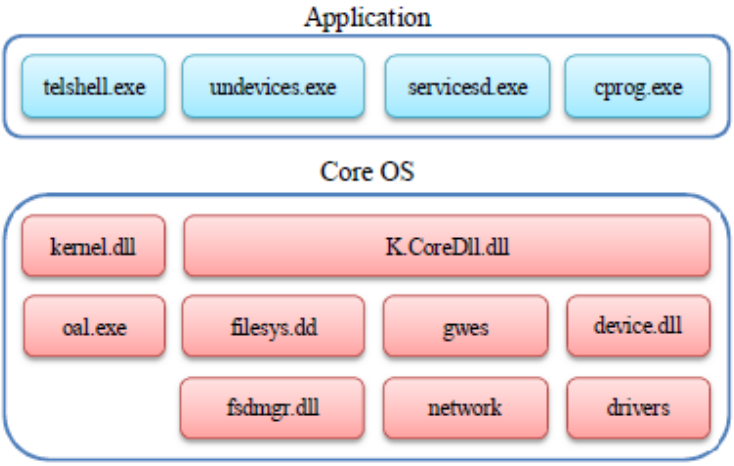
Obr. 1.3 Architektura systému iOS

Z hlediska vývoje aplikací je operační systém iOS poměrně uzavřený. Jediný způsob, jak lze do telefonu instalovat aplikace je prostřednictvím obchodu AppStore. *SDK* je tak dostupné pouze pro operační systém Mac OS X a jeho možnosti jsou podobné jako u systému Androidu, ovšem pro testování aplikací na vlastním přístroji je nejdříve nutné se zaregistrovat do placeného programu. [Tokár, 2011]

1.3 Operační systém Windows Phone 8

Vývoj tohoto systému byl započat společností Microsoft v roce 2004, ale trval příliš dlouho a tak v roce 2008 provedl Microsoft reorganizaci pracovní skupiny. Z tohoto důvodu byl projekt dočasně pozastaven a finální verze systému vyšla až roku 2010. Momentálně se tak jedná o nejmladší operační systém pro mobilní telefony na trhu. Windows Phone 8 oproti ostatním systémům z počátku nepodporoval multitasking, za což byl uživateli kritizován. Tento nedostatek byl následně prostřednictvím aktualizace odstraněn. [Tokár, 2011]

Architektura systému je rozdělena na dvě části (Obr. 1.4). Uživatelskou část, která obsahuje aplikace a služby poskytované systémem a část s jádrem systému. Jádro systému obsahuje souborový systém, ovladače, nástroje pro vykreslování grafiky, atd. Samotné jádro je založené na Windows CE 6.0, které bylo speciálně navrženo pro mobilní zařízení. [Tokár, 2011]



Obr. 1.4 Architektura systému Windows Phone 8

Pro vytváření aplikací pro platformu Windows Phone 8 má vývojář možnost volby ze dvou technologií. Microsoft Silverlight je vhodný pro vývoj běžných aplikací, naopak Microsoft XNA poskytuje nástroje pro vytváření her s vysokými nároky na grafiku. Vývojář má u obou technologií přístup k hardwarovým funkcím přístroje. Pro prodej

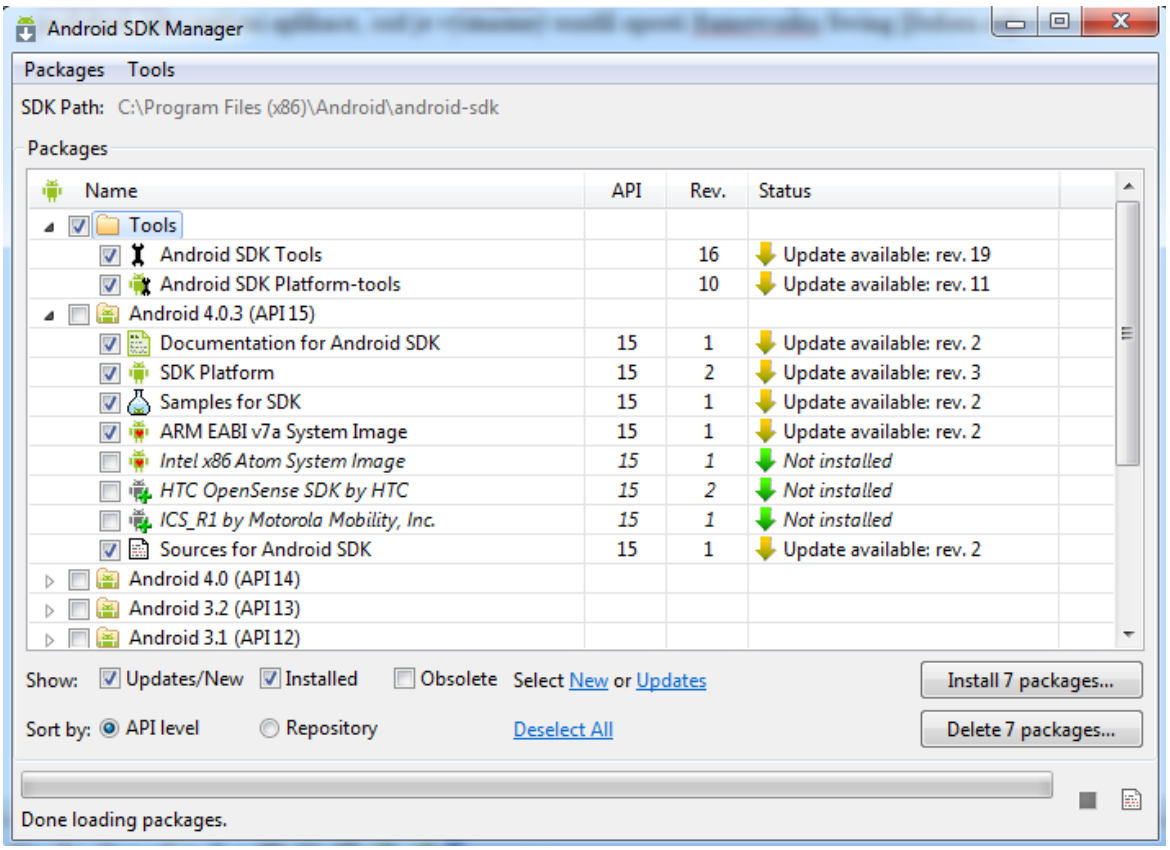
vlastních aplikací přes Marketplace je nutné zaplatit roční poplatek, který se však pro potřeby testování netýká studentů. [Tokár, 2011]

2 ECLIPSE

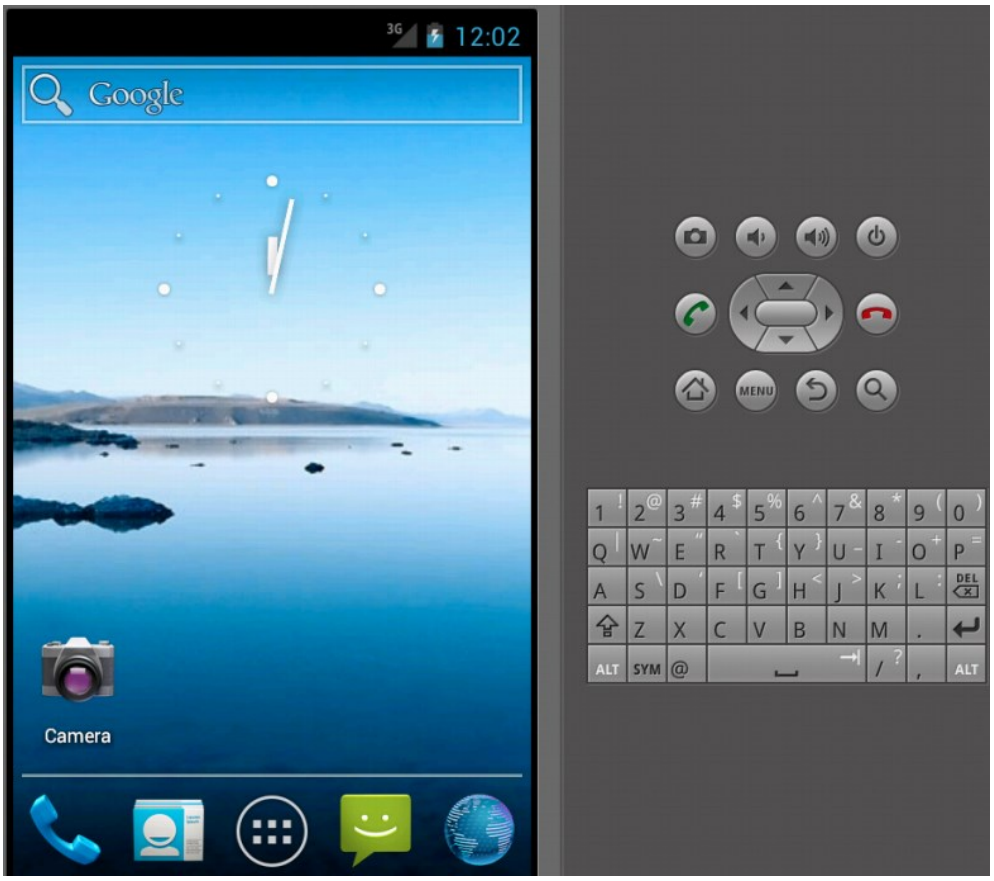
Vývoj projektu Eclipse byl zahájen na přelomu tisíciletí ve společnosti *IBM*. Jedná se o integrované vývojové prostředí, především pro vývoj programů v *Javě*, čímž je zaručena jeho poměrně snadná přenositelnost na různé platformy. Samotné jádro projektu je v porovnání s jinými vývojovými prostředími relativně malé, ovšem díky svému systému přídavných modulů (pluginů) jsou však ve skutečnosti možnosti Eclipse mnohem větší. Může posloužit pro vývoj aplikací v programovacích jazycích *C*, *C++*, *Python*, *PHP*, *Fortran* či *COBOL*. Existují pluginy pro tvorbu *XML* souborů a správu webových aplikací. Eclipse se také používá pro vývoj *RCP* aplikací, využívaných především ve vnitrofiremních projektech. Společnost *IBM* navíc v rámci vývoje platformy Eclipse iniciovala i vznik grafického frameworku *SWT*, který zaručuje, že se Eclipse bude chovat na všech operačních systémech podobně, jako nativní aplikace, což je významný rozdíl oproti frameworku Swing. [Tišnovský, 2012]

2.1 Instalace vývojového prostředí pro Android

Postup instalace se skládá z několika částí. Nejdříve je nutné ze stránky android.developers.com stáhnout Android *SDK*, který obsahuje soubor knihoven a emulátor operačního systému Android na počítači. Ten bude využit při testování aplikací. Chová se úplně stejně jako systém Android v mobilním telefonu. Takže při ladění aplikace ji není nutné nahrávat do telefonu, ale lze jí snadno otestovat přímo v počítači. Emulátor (Obr. 2.2) umožňuje výběr verze systému (Obr. 2.1), velikost displeje i virtuální paměťové karty. Po nainstalování lze instalovat aplikace, mazat je, prohlížet webové stránky, hrát hry atd. Jediné, co nefunguje je *GPS* a bezdrátové moduly Bluetooth a *Wi-Fi*. Námi vytvořené aplikace se pak zobrazují v menu, takže je možné je opakovaně spouštět přímo z emulátoru.



Obr. 2.1 Výběr verze operačního systému v *SDK* manažeru



Obr. 2.2 Emulátor systému Android

2.2 Založení nového projektu

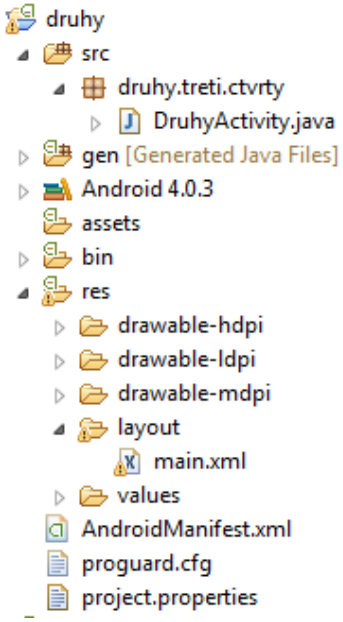
Po úspěšném nainstalování Eclipse založíme nový projekt. Z nabídky zvolíme Android Projekt a zadáme jeho jméno. V následující nabídce vybereme verzi *SDK*, pro kterou chceme programovat, *Package name* a tlačítkem *Finish* projekt vytvoříme. Na pracovní ploše vlevo se zobrazují všechny naše projekty. Ve stromové struktuře nás nejvíce zajímají adresáře *src* a *res*, se kterými budeme pracovat (Obr. 2.4).

V adresáři *res* pak podadresář *layout*, ve kterém je uložen *xml* soubor. Po jeho otevření se zobrazí displej telefonu s názvem naší aplikace a paleta nástrojů, z které si na displej můžeme přetahovat libovolné komponenty. V prostředí Eclipse je možné vytvářet tlačítka, textové pole atd. i pomocí kódu. Oba způsoby návrhu jsou spolu provázány a změna v kódu se automaticky projeví ve vizuálním návrhu a naopak. Při relativním nastavení pozice objektů na ploše si je lze libovolně přetahovat a Eclipse sám nabízí možnost zarovnání, kterou můžeme a nemusíme využít. Trvá sice mnohem déle, než se s touto vlastností vývojář naučí pracovat, ale s odstupem času určitě naopak spoustu času při vizuálním návrhu ušetří. Nebo pokud programátor již zná rozměry a pozice komponent, které bude využívat, může je vytvářet a pozicovat přímo v *xml* souboru a grafický návrh použít pouze pro kontrolu. Nejdůležitější částí je zdrojový kód aplikace (Obr. 2.3), který se zpravidla nachází v adresáři *src*. Ve stromové struktuře na (Obr. 2.4) má jméno *DruhyActivity.java*.

Třída *DruhyActivity* (Obr. 2.3) je založena na třídě *Activity*, což je jednoduchá entita aplikace určená k provádění akcí. Jakmile aktivita startuje, je volána metoda *onCreate()*, ve které nalezneme kód. Pro zobrazení textu je použita podtřída *TextView*, do které je vepsán text „Monika“. Důležité je nezapomenout na import balíčku *android.widget.TextView* s kterým pracujeme. Program lze následně zkompilovat a spustit pomocí klávesy F11.

```java
package druhy.treti.ctvrty;

import android.app.Activity;


public class DruhyActivity extends Activity
{
    /** Called when the activity is first created. */
    TextView text;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        text= (TextView) findViewById(R.id.textView1);
        text.setText("Monika");
    }
}
```

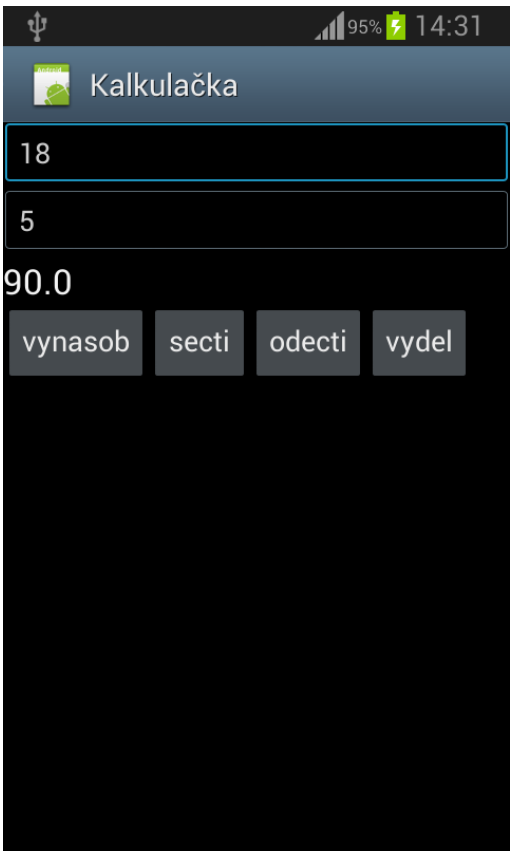
Obr. 2.3 Zdrojový kód aplikace



Obr. 2.4 Stromová struktura s adresáři

2.3 Ukázková aplikace v prostředí Eclipse

Z důvodu názornosti byla vytvořena aplikace (Obr. 2.5), která po zadání dvou čísel provede požadovanou matematickou operaci. Pro tento účel si vystačíme se základními matematickými operacemi sčítání, odčítání, násobení a dělení. V Eclipse je potřeba otevřít adresáře *res* a *layout*, ve kterém je umístěn soubor *main.xml*. Po jeho otevření se zobrazí okno s paletou nástrojů, ze které si na obrazovku přetáhneme prvky *Button*, který slouží pro výpočet, *editText* pro zadávání čísel a *textView* pro zobrazení výsledku.



Obr. 2.5 Obrazovka aplikace Kalkulačka

V adresáři *src* si otevřeme systémem vytvořený *soubor.java*, ve kterém jsou nadefinovány všechny použité prvky a ke každé matematické operaci je založena třída (Obr. 2.6). Každá třída má svůj název, přes který je volána při stlačení příslušného tlačítka pro výpočet. V každé třídě se pak nachází deklarace proměnné a výpočet samotný. Ve funkci *OnClickListener* jsou už jen zadávané hodnoty převedeny na *String*, zavolána příslušná třída a vypsán výsledek do výstupu. Pro testování aplikace byla použita nejnovější verze emulátoru Android 4.0.3.

```
package kol.lok.lko;
public class operace
{
    double soucet(double a , double b)
    {
        double s = a + b;
        return s;
    }

    public operace() {
        // TODO Auto-generated constructor stub
    }
}
```

Obr. 2.6 Třída operace pro součet dvou proměnných

Snímek obrazovky aplikace (Obr. 2.5) byl pořízen mobilním zařízením Samsung Galaxy SII (Obr. 2.7), pro které byla dále vyvíjena aplikace na ovládání modelu robotu. Výhodou oproti emulátoru je možnost využití bezdrátových technologií Bluetooth a *Wi-Fi*. Pokud je zařízení v době kompilace v prostředí Eclipse připojeno k *PC*, vývojové prostředí zobrazí dotaz, na kterém zařízení má být aplikace spuštěna. V případě, že není připojen mobilní telefon, automaticky dojde ke spuštění aplikace v emulátoru. Pokud je mobilní zařízení připojeno a vybráno pro spuštění aplikace, je tato do něj automaticky nainstalována a následně spuštěna.

Obr. 2.7 Mobilní telefon Samsung Galaxy SII [Srb, 2011]

3 KONSTRUKCE MODELŮ

Úkolem mé diplomové práce bylo vytvořit aplikaci pro ovládání reálného modelu. Bylo tak nutné vybrat zařízení, které bude schopné pohybu a zároveň také komunikace s mobilním zařízením. K tomuto účelu mi byla zapůjčena stavebnice Lego Mindstorms. Pro srovnání byla použita robotická koule Sphero.

3.1 Lego Mindstorms

Lego Mindstorms je programovatelná robotická stavebnice vyráběná firmou Lego. Stavebnice ve verzi Education (Obr. 3.1), která mi byla k dispozici, obsahuje LEGO NXT programovatelnou kostku s Bluetooth, tři servomotory, dva tlakové senzory, světelný, zvukový, ultrazvukový senzor, kabeláž a sadu stavebních dílů.

Ke stavebnici je dodáván programovací software NXT-G, který reprezentuje vizuální programování. Toto prostředí obsahuje pouze základní funkce a je tak vhodné spíše pro začátečníky. Z tohoto důvodu bylo použito vývojové prostředí Bricx, založené na syntaxi jazyka *C*. V následujících podkapitolách jsou blíže popsány funkční prvky stavebnice, které byly využity pro stavbu robotu. Poslední podkapitola je pak věnována konstrukci modelu robotu.

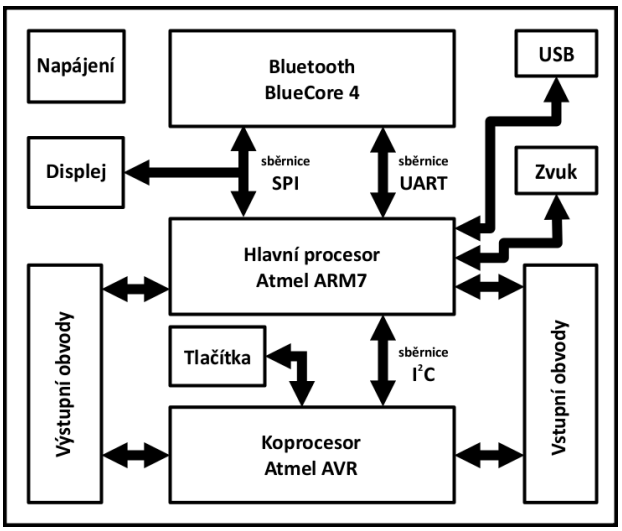
Obr. 3.1 Lego Mindstorms – verze Education [Eduxe, 2013]

3.1.1 Programovatelná jednotka NXT

Hlavním prvkem stavebnice je programovatelná 32-*bit*ová jednotka s Bluetooth bezdrátovou komunikací a portem *USB* 2.0, která je vybavena 4 vstupními a 3 výstupními porty, displejem o rozlišení 60x100*px* a 8K*Hz* reproduktorem. K napájení je použit 9*V* akumulátor. Jednotka je osazena 32-*bit*ovým mikroprocesorem ARM7 s 256 *KB flash* pamětí, 64 *KB* pamětí *RAM* a 8-bitovým koprocesorem. Připojená zařízení komunikují pomocí 6ti vodičového konektoru *RJ12*. Na (Obr. 3.3) je znázorněno schéma propojení jednotlivých bloků v řídicí jednotce. [LEGO Group, 2006]

Obr. 3.2 Řídicí jednotka NXT [Eduxe, 2013]

Do paměti jsou kromě firmwaru ukládány programy, obrázky a zvuky. *Flash* paměť je přepisovatelná, takže po jejím zaplnění stačí nepoužívaná data smazat a nahradit je novými. Programovatelná jednotka NXT je řízena firmwarem, a to i bez našeho vlastního spuštěného programu, kterému pouze na určitou dobu předává řízení nad připojenými zařízeními. Tento řídicí program je důležité udržovat aktuální, protože každá aktualizace opravuje nalezené chyby a rozšiřuje funkci zařízení. Jedním u úkolů firmwaru je komunikace s programovacím jazykem, takže každý programovací jazyk má většinou svou verzi firmwaru, kterou je potřeba do zařízení nahrát. [Jakeš, 2013; LEGO Group, 2006]



Obr. 3.3 Propojení bloků v řídicí jednotce [Jakeš, 2013]

3.1.2 Servomotory

Pohyb robotu zajišťuje servomotor. Každý motor má vestavěný senzor rotace, což umožňuje provádět řízení s přesností na jeden stupeň. Senzor měří otáčky ve stupních, nebo v plných obrátkách, přičemž jedna obrátka je rovna 360°. Motory se můžou pohybovat rozdílnou rychlostí, ale existují i funkce pro jejich synchronizaci.

Obr. 3.4 Servomotor [Eduxe, 2013]

Pohyb motoru lze realizovat několika způsoby. Nejpřesnější z nich je pomocí nastavování stupňů v rozsahu 0 až 360°. Tímto motoru udáváme, o kolik stupňů se má otočit. Úhel natočení je motorem zajišťován zpětně přes počítadlo otáček. Protože se ale nejedná o krokový motor, není pohyb zcela přesný a je tak nutné počítat s odchylkou v řádu jednotek. Další způsob je nastavováním počtu otáček, tzn. kolikrát se motor otočí o 360°. Tento způsob zejména při bezdrátové komunikaci tolik nezatěžuje řídicí jednotku, ale vykazuje mnohem nižší přesnost.

Další dva způsoby otáčení motoru jsou vázány na čas. Zde můžeme v sekundách nastavit dobu otáčení motoru, nebo použít funkci, která bude motorem otáčet do té doby, dokud program neskončí. Posledním parametrem, který je pro všechny funkce společný, je nastavení výkonu motoru [LEGO Group, 2006].

3.1.3 Ultrazvukový senzor

Ultrazvukový snímač umožňuje detekovat objekty, pohyb a měřit vzdálenost. Snímač vysílá cyklicky vysokofrekvenční impuls. Pokud se impuls odrazí od předmětu a vrátí se zpět přijímači, vyhodnocovací elektronika senzoru vypočte časový interval mezi vyslaným a přijatým signálem a určí vzdálenost od objektu.

Použitý ultrazvukový senzor umožňuje měření vzdálenosti v centimetrech a palcích. Rozsah měření je 0 až 255 centimetrů s přesností +-3*cm*. Aby se vyslaný impuls vrátil zpět, nesmí být těleso, od kterého se impuls odráží vůči snímači pod větším úhlem než 15°. Záleží také na vzdálenosti mezi snímačem a tělesem. Čím blíže je senzor tělesu, tím je měření přesnější. Optimální je mít rozměrné objekty s tvrdým povrchem, od kterých se impuls dobře odráží, na rozdíl od předmětů z měkké tkaniny, nebo různě zakulacených, které se obtížně detekují. Při použití dvou, nebo více ultrazvukových senzorů, pracujících ve stejné místnosti může také docházet k jejich vzájemnému ovlivňování.
[LEGO Group, 2006]

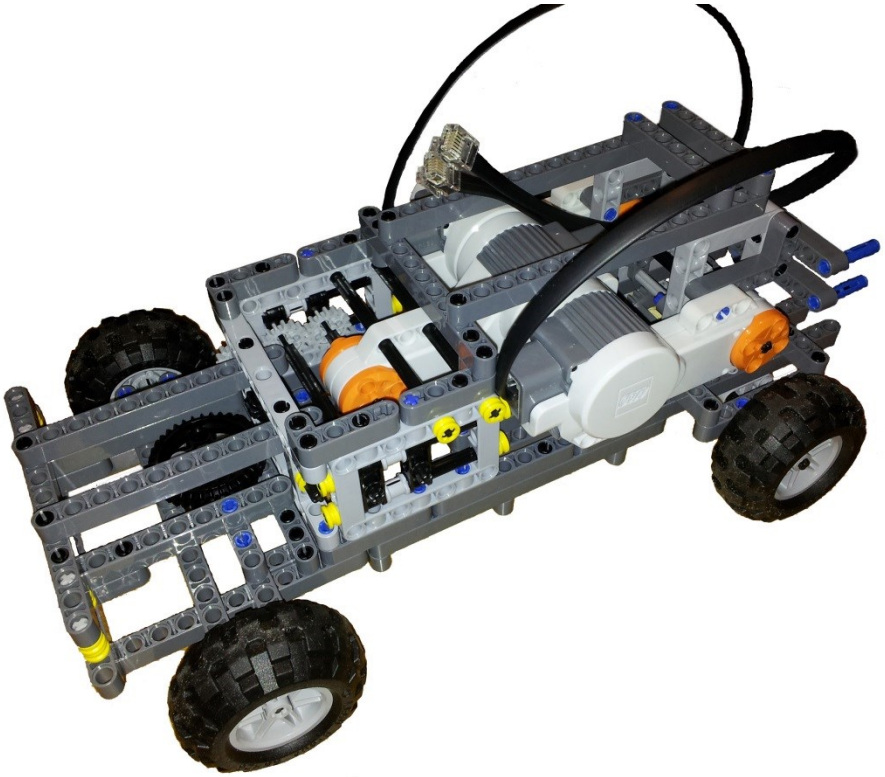
Obr. 3.5 Ultrazvukový senzor [Eduxe, 2013]

3.1.4 Konstrukce modelu robotu

Po seznámení se s možnostmi stavebnice a zprovozněním komunikace mezi jednotkou NXT a počítačem bylo nutné vytvořit model, který bude ovládán pomocí mobilní aplikace. Stavebnice obsahuje sadu kol i pásů, takže bylo možné si vybrat, jestli podvozek modelu bude diferenciální, nebo na stejném principu jako u automobilu. Nakonec byl

zvolen model automobilu (Obr. 3.6) a to i proto, že stavebnice obsahuje diferenciál, takže nebude docházet ke smýkání při jízdě do zatáčky. Navíc bylo možné vytvořit konstrukci, kde jsou pro pohon zadních kol použity dva servomotory. Toto provedení bylo zvoleno i z toho důvodu, že NXT jednotka s akumulátorem váží 0,3kg a při jízdě po povrchu s větším třecím odporem by tak jeden servomotor pro pohon zadních kol s ohledem na celkovou hmotnost modelu byl nedostačující. Nastal však problém s převodem, který byl realizován pomocí soustavy ozubených kol, které při prudším rozjezdu začaly přeskakovat. Tento problém byl vyřešen vystužením vodících tyčinek, na nichž byla ozubená kola nasazena.

Problém s ozubením nastal také u přední nápravy. V tomto případě pootočení servomotoru o pět stupňů bylo využito jen na vůli v ozubení. Tato vůle se projevovala i při najetí modelu na nerovnost, kdy vůle v řízení způsobovala samovolné zatáčení kol a znemožňovala tak ovládání celého modelu. Z tohoto důvodu byl model konstrukčně upraven a servomotor, který má za úkol zatáčet předními koly umístěn tak, aby mohl být s řízením propojen napřímo bez jakéhokoli ozubení. Ani tato úprava se neosvědčila, protože servomotor má už z výroby vůli +-3 stupně na jednu otáčku. Určitá vůle se také projevovala i při natočení o 45°, kdy byly kola modelu v přímém směru, následně byly otočeny do levé, nebo pravé úvratě a vráceny zpět.



Obr. 3.6 Návrh modelu automobilu

Kvůli těmto problémům tak bylo od modelu automobilu upuštěno a byl postaven model robotu (Obr. 3.7) s diferenciálním podvozkem a pásy. Pohon robotu mají na starost dva servomotory, přičemž každý ovládá jeden pás. Nad nimi je umístěna NXT jednotka a v přední části robotu ultrazvukový snímač, který měří aktuální vzdálenost od překážky a NXT zařízení tuto informaci posílá zpět aplikaci v mobilním zařízení. Aby bylo ovládání co nejpřesnější, byl také minimalizován počet ozubených kol. Každý z motorů pak každých 25*ms* dostává informaci o tom, jakou rychlostí a jakým směrem se má otočit o jeden stupeň. Zatáčení je tak řešeno zpomalením jednoho z motorů a jede-li robot přímo, hodnoty rychlosti na obou motorech jsou totožné. Pro otáčení na místě jsou rychlosti obou motorů stejné, ale navzájem opačně orientované.

Obr. 3.7 Finální verze modelu robotu

3.2 Sphero

Jedná se o robotickou kouli, kterou lze v současné době ovládat pomocí zařízení s operačním systémem iOS, nebo Android. Komunikace probíhá pomocí technologie Bluetooth, takže kouli je možné ovládat na vzdálenost až 15 metrů. Průhledný vnější obal je vyroben z tvrzeného polykarbonátu. Koule se dostává do pohybu na základě změny těžiště, kterou zajišťují dvě malá gumová kola. Prostřednictvím zabudovaného akcelerometru a gyroskopu má mobilní zařízení k dispozici údaje o aktuálním náklonu a rychlosti. Koule může na rovném povrchu, kde nedochází ke smýkání dosáhnout

maximální rychlosti až 3 *m/s*. Díky vícebarevným *LED* diodám je schopna měnit svou barvu, čehož využívají některé aplikace, kterých je v současnosti více než 20. Pro hry je také využíváno senzorů v mobilním zařízení, kde je využit gyroskop pro golf, kuželky nebo akcelerometr v robotické kouli, kde se mění barvy a hráč musí na určitou barvu kouli uchopit a měří se reakční doba každé osoby. O pohon se stará lithium-polymerový akumulátor o kapacitě 5000 m*Ah*, jehož nabíjení je prováděno indukčně, takže nejsou potřeba žádné konektory a koule je tak zcela voděodolná. [HALL-GEISLER, 2012]

Obr. 3.8 Robotická koule Sphero [Gosphero, 2013]

Pro vývojáře je k dispozici *SDK*. Stačí si tak nainstalovat Eclipse a pomocí *SDK* Manageru do něj nainstalovat *SDK* pro Sphero stejně, jako tomu bylo u Androidu. Programy je možné vytvářet pro Android zařízení od úrovně *API* 8, čemuž odpovídá verze Android 2.2. Zařízení se starší verzí systému nejsou podporována. Na stránkách výrobce je k dispozici několik ukázkových příkladů. [www.gosphero.com]

4 APLIKACE PRO OVLÁDÁNÍ MODELU ROBOTU

Po vytvoření modelu mobilního robotu byl další částí práce návrh a realizace mobilní aplikace. Nejdříve bylo nutné zprovoznit bezdrátovou komunikaci mezi modelem a mobilním zařízením. K tomuto účelu byla pro testování vytvořena aplikace v prostředí jazyka C#, ve které bylo dále navrženo uživatelské rozhraní aplikace. Po úspěšném odzkoušení proběhla implementace mobilní aplikace do prostředí jazyka Java.

4.1 Komunikace mezi modelem a mobilním zařízením

Pro komunikaci mezi modelem a mobilním zařízením je možné využít technologii Bluetooth, nebo *Wi-Fi*. Co se týče přenosových rychlostí, komunikace přes Bluetooth, která v asynchronním režimu zvládne 720 *kbps* je oproti *Wi-Fi* s 11 *Mbps* o několik řádů pomalejší. *Wi-Fi* má také větší dosah, ale to pro účel této práce není až tak důležité. Naopak, protože Bluetooth má nižší požadavky na napájení než *Wi-Fi*, z toho důvodu je také mnohem menší a je tak součástí NXT kostky. Protože externí *Wi-Fi* modul nebyl součástí stavebnice, která mi byla zapůjčena, byla v této práci pro komunikaci mezi mobilním zařízením a modelem robotu využita technologie Bluetooth.

Komunikace na straně NXT jednotky byla vytvořena ve vývojovém prostředí Bricx, založeném na syntaxi jazyka C. Na (Obr. 4.1) je část kódu vytvořené komunikace v tomto programovacím jazyce, resp. funkce *BTSendMessage*, která je naprogramována následujícím způsobem. Před zahájením jakékoli komunikace je nutné zkontrolovat, zda jsou zařízení Bluetooth v NXT kostce a mobilním zařízení navzájem připojena. Toto zajišťuje funkce *BTCommCheck*. Jestliže spustíme program v NXT a tato funkce nevrací *NO_ERR*, spojení není navázáno. Na displej NXT je pak vypsáno "Bluetooth nebylo připojeno" a zařízení zároveň vydává přerušovaný tón. V případě, že je spojení navázáno, je zařízení připraveno pro příjem (*ReceiveMessage*), resp. odesílání dat (*BTSendMessage*).

Pro posílání a přijímání dat z externího zařízení musí být dodržen následující formát. První pole (*__buffer[0]*) určuje, zda je vyžadována odpověď. V tomto případě hodnota 0x80 znamená, že není. Následující hodnota 0x09 je příkaz pro zápis. Pole *__buffer[2]* je tzv. *mailbox*, což je schránka, ve které jsou proměnné, jako je rychlost a vzdálenost. V posledním poli je uložena celková velikost zprávy. Všechny konstanty i proměnné v řetězci musí být typu bajt.

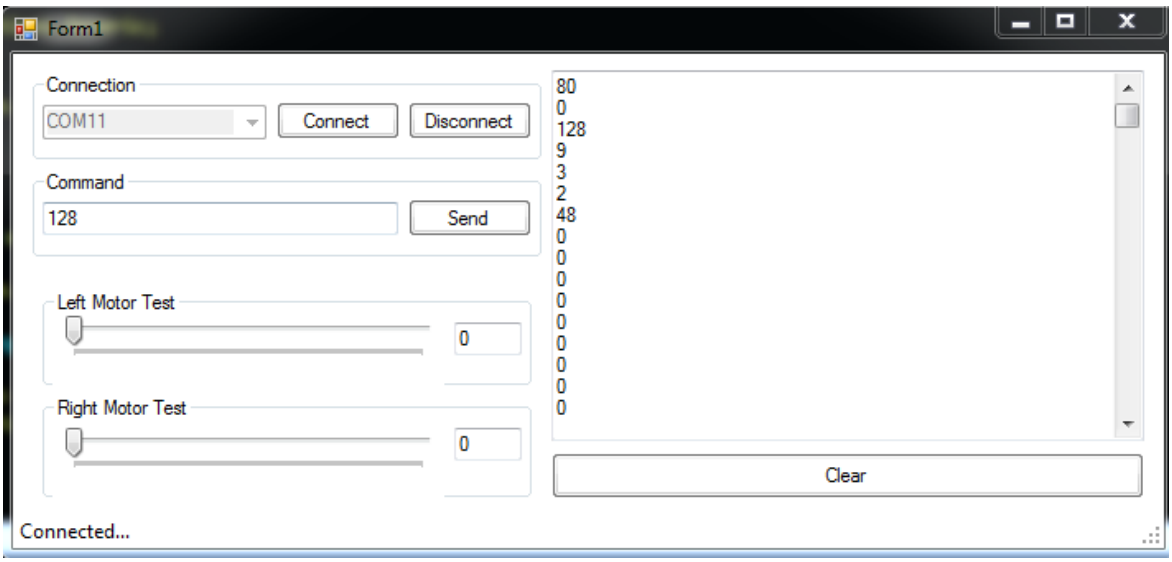
```
1  sub BTSendMessage(byte connection, byte mailbox, string msg)
2  {
3     byte len,Smb;
4     int i;
5     BTWait(connection);
6     ArrayInit(__buffer,0,80);
7     ArrayInit(__array,0,59);
8     StrToByteArray(msg,__array);
9     len = ArrayLen(__array);
10
11    __buffer[0] = 0x80;                    //bez odpovědi
12    __buffer[1] = 0x09;                    //příkaz pro zápis
13    __buffer[2] = mailbox;                 //mailbox
14    __buffer[3] = len+1;                   //velikost zprávy
15
16    for (i=4;i<(len+4);i++){
17       __buffer[i] = __array[i-4];
18 }
```

Obr. 4.1 Komunikace v prostředí jazyka Bricx

4.2 Testovací aplikace v prostředí jazyka C#

Hlavním důvodem pro vytvoření testovací aplikace v prostředí jazyka C# bylo zprovoznění komunikace mezi jednotkou NXT a *PC*. Protože tato část byla časově náročná, nebylo proto vhodné testovat komunikaci v prostředí jazyka Java, kde je aplikace vytvářena na *PC*, zkompilována a následně nahrána do mobilního zařízení. Z tohoto důvodu byl využit *PC* se zabudovaným Bluetooth zařízením.

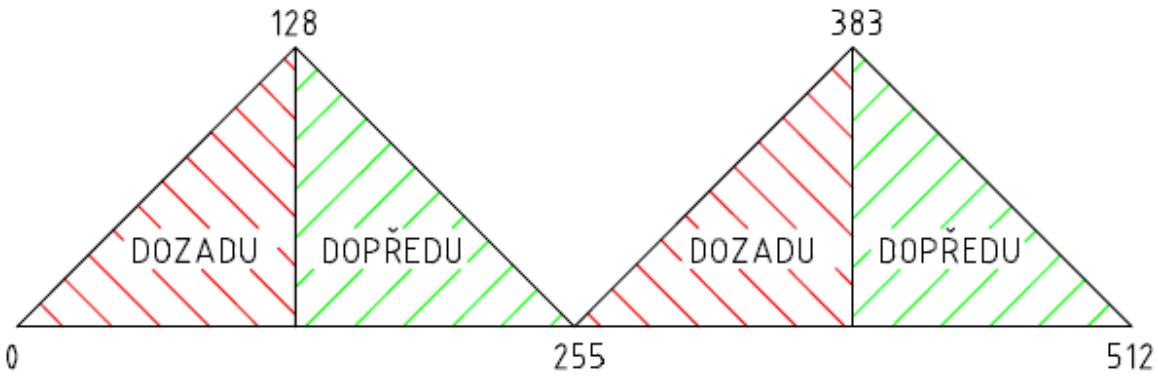
Testovací aplikace (Obr. 4.2) nejprve obsahovala pouze pole pro výběr sériového portu, tlačítko pro připojení/odpojení od zařízení a textové pole, do kterého byl vypsán přijatý řetězec z NXT. Po zprovoznění komunikace z NXT do *PC* bylo do testovací aplikace přidáno textové pole a tlačítko pro odeslání řetězce znaků, který byl do tohoto pole vepsán. Řetězec přijatý NXT zařízením byl následně pomocí funkce *TextOut* vypsán na display.



Obr. 4.2 Testovací aplikace

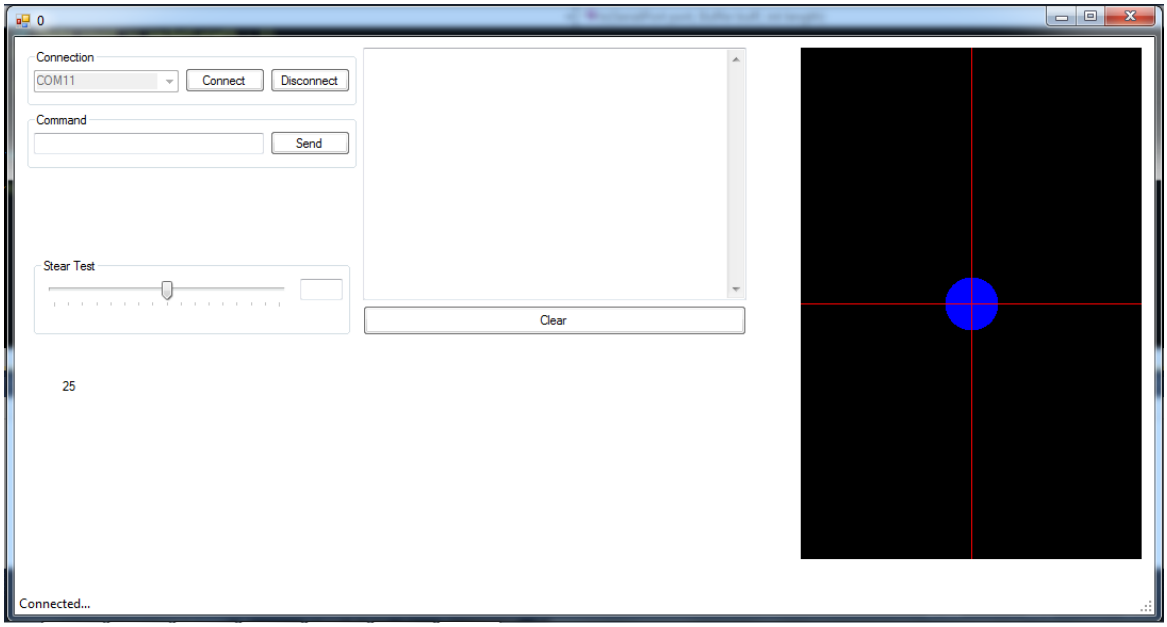
Po zprovoznění obousměrné komunikace byl k NXT jednotce připojen servomotor. V prostředí *C#* byl vytvořen posuvník, který je navázán na proměnnou *speed*. Rozsah byl nastaven na hodnotu 0-255. V prostředí Bricx byla pro příjem dat z aplikace v *PC* použita funkce *OnFwd*(*OUT_B*, *speed*). Tato funkce má dva parametry. První parametr je nastavení výstupního portu, ke kterému je motor připojen. V tomto případě port B. Dalším parametrem je rychlost, kterou nastavujeme pomocí posuvníku (proměnná *speed*).

Funkce *OnFwd* funguje tak, že otočí motorem o jeden stupeň požadovanou rychlostí. Experimentálně bylo zjištěno, že hodnoty 0-128 jsou určeny pro otáčení motoru dopředu a hodnoty 129-255 opačným směrem. V krajních bodech, tj. 0 a 255 je motor v klidu. Nicméně u modelu robotu je z důvodu konstrukce směr pohybu robotu opačný vzhledem ke smyslu otáčení motoru, což je znázorněno na (Obr. 4.3). Pro hodnoty 255-512 se robot chová analogicky.



Obr. 4.3 Směr pohybu robotu

Dále byl v prostředí jazyka C# vytvořen návrh obrazovky aplikace, pomocí které bude robot ovládán (Obr. 4.4). Jde o trackball, který je po spuštění aplikace umístěn uprostřed obrazovky. V této pozici je robotu posílána rychlost *speed* = 255. Jak jde vidět v grafu na (Obr. 4.3) při nastavení hodnoty rychlosti na 255 robot stojí. Maximální rychlosti ve směru dopředu je dosaženo, je-li trackball po svislé ose tažen až na okraj horní části obrazovky. V tento okamžik je rychlost *speed* = 128. Pro dosažení maximální rychlosti ve směru dozadu je hodnota rychlosti *speed* = 383 a pozice trackballu na obrazovce je na svislé ose u spodního okraje. Rozsah hodnot 128-383 pro rychlost byl použit z důvodu plynulého rozjezdu robotu z klidového stavu. Zatáčení robotu bylo zkušebně realizováno zastavením jednoho z motorů.



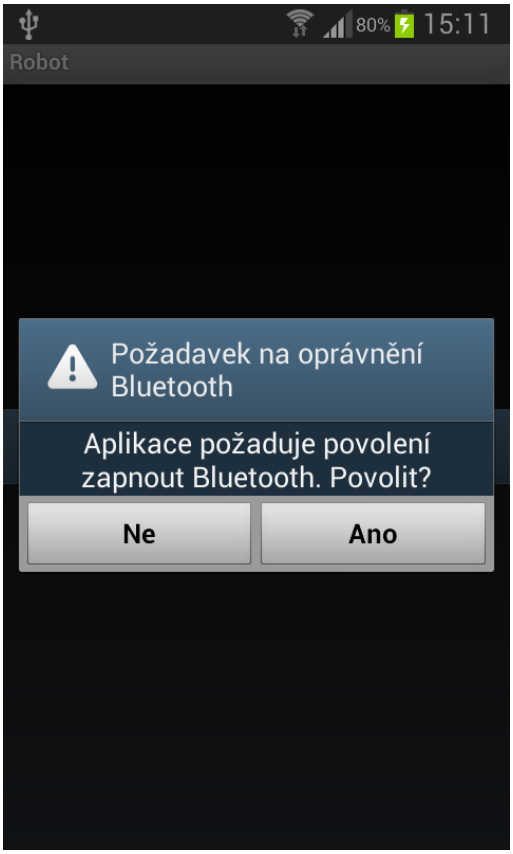
Obr. 4.4 Testovací aplikace s návrhem obrazovky

4.3 Mobilní aplikace v prostředí jazyka Java

Pro vytvoření mobilní aplikace pro ovládání modelu robotu bylo nutné veškerou komunikaci mezi NXT zařízením a *PC* přepsat do jazyka Java. Na základě návrhu bylo vytvořeno uživatelské rozhraní aplikace, které bylo doplněno o menu a lištu v horní části obrazovky. Po připojení k NXT zařízení je model robotu možné ovládat interakcí s displejem, nebo prostřednictvím hlasového vstupu. Pro ovládání hlasem je nezbytné připojení k internetu a to z toho důvodu, že pro verzi systému Android 4.0.3, pro kterou je aplikace zkompilována probíhá rozpoznávání hlasu online.

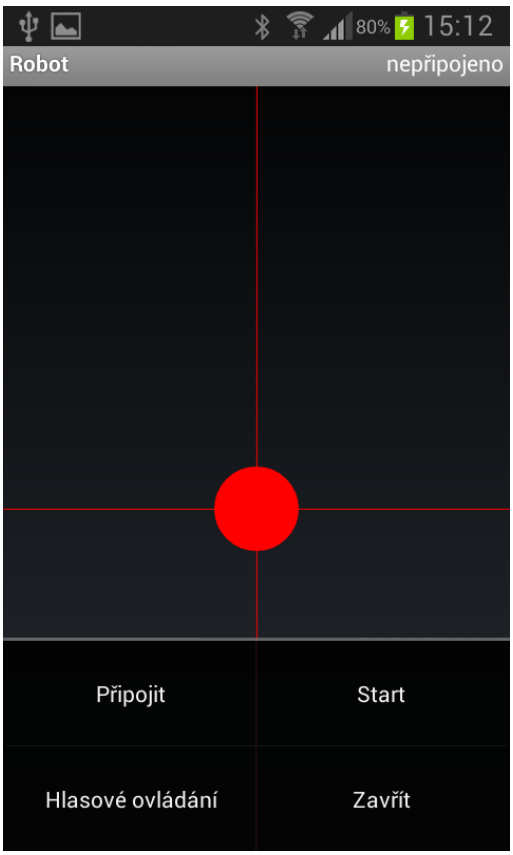
4.3.1 Uživatelské rozhraní

Po spuštění aplikace je nejprve kontrolováno, zda je zapnut adaptér Bluetooth (Obr. 4.5). V případě, že je vypnutý, je uživatel aplikací požádán o oprávnění jej zapnout. Výběrem možnosti *Ano* dojde k zapnutí Bluetooth a spustí se uživatelské rozhraní aplikace. V případě, že jeho zapnutí není povoleno, aplikace je ukončena. Pokud je Bluetooth zapnutý už před spuštěním samotné aplikace, tato výzva se uživateli nezobrazuje a dojde k okamžitému spuštění aplikace.



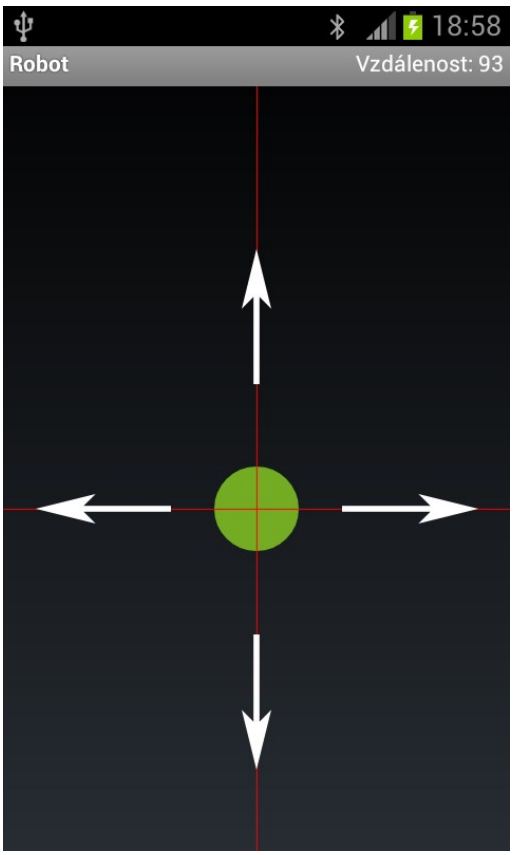
Obr. 4.5 Požadavek na oprávnění Bluetooth

V horní části obrazovky (Obr. 4.6) je na liště název aplikace a informace o tom, že zařízení není připojeno. Tento stav také signalizuje červeně zbarvený trackball umístěný uprostřed obrazovky, jehož tahem se robot ovládá. Zároveň je znemožněno s ním jakkoli pohybovat. Je tak nutné vyvolat kontextové menu, zvolit možnost *Připojit* a následně vybrat zařízení pro připojení. V případě, že zařízení není v seznamu, je nutné zvolit možnost *Vyhledat zařízení* a provést spárování. Pokud spárování proběhne úspěšně, při dalším připojení k zařízení již toto zařízení bude v seznamu spárovaných zařízení a nebude nutné jej znovu vyhledávat.



Obr. 4.6 Uživatelské rozhraní aplikace s menu

Po úspěšném připojení k robotu je tento stav zobrazen v horní liště aplikace a barva trackballu je nyní oranžová. Pokud je program v robotu spuštěn, je možné v mobilní aplikaci v menu pomocí tlačítka *Start* spustit vlákno pro příjem a vlákno pro odesílání dat z mobilního zařízení. Tento stav je signalizován zbarvením trackballu na zeleno. V horní liště je následně místo stavu připojení zařízení zobrazována vzdálenost robotu od překážky (Obr. 4.7). Při interakci s displejem trackball změní barvu na modrou, což signalizuje odchozí komunikaci a robot se pohybuje ve zvoleném směru. V případě, že se přeruší dotyk s displejem, trackball se vrátí do původní polohy a robot se zastaví. Aplikaci lze ukončit pomocí tlačítka *Zavřít* v menu. V případě, že bylo spojení přes Bluetooth přerušeno, mobilní aplikace vypíše na display „Připojení k zařízení bylo ztraceno“ a ukončí se.



Obr. 4.7 Mobilní aplikace v chodu

4.3.2 Rozlišení displeje

Aplikace byla navržena tak, aby se dokázala přizpůsobit zařízením s různým rozlišením displeje. K tomuto účelu byla využita funkce *getDisplayResolution*, ve které je použita metoda *getSize*, která vrací velikost displeje v pixelech (Obr. 4.8). Rozlišení displeje je uloženo ve formátu šířka x výška a následně použito pro určení středu a vykreslení trackballu. Získané rozlišení displeje je dále použito ve třídě *drive*, kde je řešeno ovládání motorů v závislosti na poloze trackballu na displeji mobilního zařízení. Tento krok je důležitý z toho důvodu, aby byla pro ovládání modelu robotu využita vždy celá plocha displeje. Protože v aplikaci je povolena pouze orientace na výšku, nemůže při výpočtu rozlišení displeje dojít k záměně parametrů a chybnému výpočtu.

```
private Point getDisplayResolution()
{
    Display display = getWindowManager().getDefaultDisplay();
    Point size = new Point();
    display.getSize(size);
    return size;
}
```

Obr. 4.8 Výpočet rozlišení displeje

Metodu *getSize* je možné použít pro operační systém Android 4.0.3 a vyšší. Protože mi bylo k dispozici pouze jedno zařízení s touto verzí systému, aplikace byla testována pouze pro displej s rozlišením 480 x 800 *px*. Pro zařízení s jiným rozlišením displeje proběhne vykreslení grafické části korektně, ale třída *drive* již nebyla vytvořena, protože by nebylo možné otestovat její funkčnost.

4.3.3 Ovládání modelu

V prostředí jazyka Bricx bylo nutné zajistit, aby motory dokázaly pracovat paralelně. Zatáčení modelu robotu bylo dosud řešeno zastavením jednoho z motorů, což neumožňovalo plynulé projetí zatáčkou. Model robotu se tak dokázal pohybovat plynule pouze v přímém směru a v okamžiku, kdy byly rychlosti, které motory zpracovávají rozdílné, nedokázala je řídicí jednotka zpracovat paralelně.

Řešení tohoto problému je na (Obr. 4.9). Funkce *setRSpeed* obsluhuje pravý motor robotu, kde přijatý řetězec v *mailbox*u převede na číslo a uloží jej do proměnné *speed*. Vstupem do funkce *OnFwd* je port na řídicí jednotce, ke které je motor připojen a proměnná *speed*. Funkce *OnFwd* je aktivní po dobu 25*ms*. Za stejný časový úsek je z mobilní aplikace vyslán nový řetězec. Funkce pro levý motor je totožná, ale proměnná speed má zápornou hodnotu, což mění smysl otáčení motoru. Tato úprava byla provedena z konstrukčního důvodu, protože motory jsou umístěny proti sobě a v případě, že by nebyla provedena, robot by se při stejné hodnotě rychlosti pro oba motory otáčel na místě.

Aby Funkce *setRSpeed* a *setLSpeed* byly vykonávány současně, musí být uvozeny synchronizační funkcí *Acquire* a po jejich použití zakončeny funkcí *Release*. Ve funkci *main* je nutné použít funkci *Precedes*, která zajistí, aby úkoly byly prováděny současně, pokud tomu nebrání jiné závislosti. Vstupní parametry této funkce jsou funkce, které mají být vykonávány současně. V tomto případě je to funkce *drive* a *distance*. Ve funkci *distance* je realizováno odesílání vzdálenosti robotu od překážky.

```
void setRSpeed(byte connection, byte mailbox, byte flush)
   {
        string  speedMsg = BTReceiveMessage(connection, mailbox, flush );
        int speed = StrToNum(speedMsg);
        OnFwd(OUT_B, speed);
        Wait(25);
        Off(OUT_B);
   }

task drive()
   {

  while(true)
      {
         Acquire(moveMutex);
           setLSpeed( BT_CONN, 2, true );
           setRSpeed( BT_CONN, 1, true );
         Release(moveMutex);
      }
   ...
  }
```

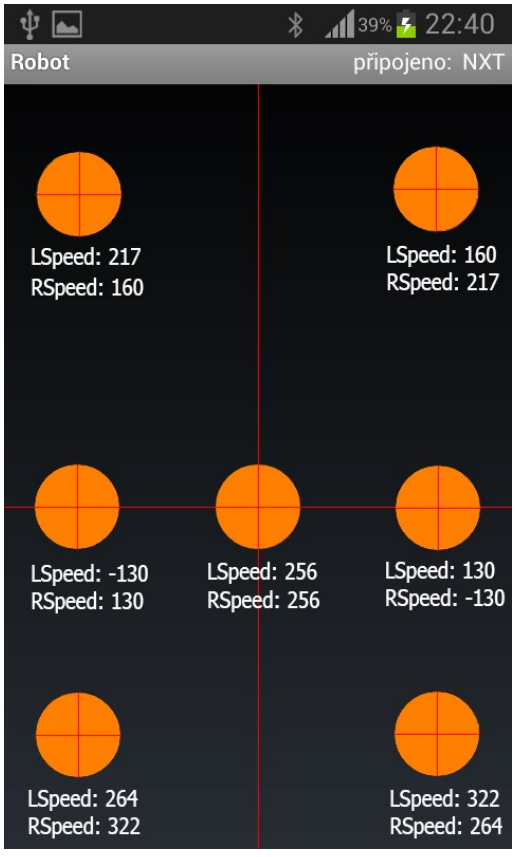
Obr. 4.9 Zajištění paralelního chodu motorů

Na straně mobilní aplikace byla vytvořena třída *drive*, ve které je řešeno ovládání mobilního robotu. Pro rozlišení displeje 480 x 800 *px* byla pro jízdu dopředu proměnná *speed* omezena na maximální hodnotu *speed* = 130. Pro pohyb opačným směrem je maximální hodnota rychlosti *speed* = 357. Zatáčení robotu bylo realizováno zpomalením jednoho z motorů. Toto zpomalení je vypočteno z proměnné *Value*, která je pro jízdu přímo rovna 0. Pro zatočení modelu robotu vlevo nabývá proměnná *Value* hodnot v rozsahu od -1 do -78. Pro opačný směr je tento rozsah 1 – 78. Tato proměnná tak s rostoucí vzdáleností od vertikální osy na displeji aplikace roste. Na základě proměnných *speed* a *Value* jsou vypočteny rychlosti pro levý *(LSpeed*) a pravý (*RSpeed*) motor. Tyto rychlosti jsou následně odeslány do NXT zařízení.

Ukázka výpočtu pro pohyb modelu robotu v přímém směru, pro plynulý průjezd levotočivou zatáčkou a otáčení robotu na místě je uvedena na (Obr. 4.10). Různé polohy trackballu s rychlostmi, které se řídicí jednotce při dané poloze odesílají, jsou znázorněny na (Obr. 4.11). Pro otáčení modelu robotu na místě byly použity stejné rychlosti pro oba motory. Podle směru otáčení je vždy jedna z hodnot opačná. Robot se otáčí na místě, jestliže proměnná *speed* nabývá hodnot v rozsahu 245-265 a zároveň hodnota proměnné *value* není v intervalu -10-10. Rychlost otáčení je dána aktuální hodnotou proměnné *value*.

```
//rovno
    if (value == 0)
    {
        LSpeed = Speed;
        RSpeed = Speed;
    }

//dopredu_doleva
    if (value < 0)
    {
        LSpeed = Speed - value;
        RSpeed = Speed;
    }

//na miste1
    if ((Speed >= 245) && (Speed <= 265) && (value >= 50))

    {
        LSpeed = 130;
        RSpeed = -130;
    }

    if ((Speed >= 245) && (Speed <= 265) && (value <= -50))

    {
        LSpeed = -130;
        RSpeed = 130;
    }
```

Obr. 4.10 Ukázka výpočtu rychlostí pro motory

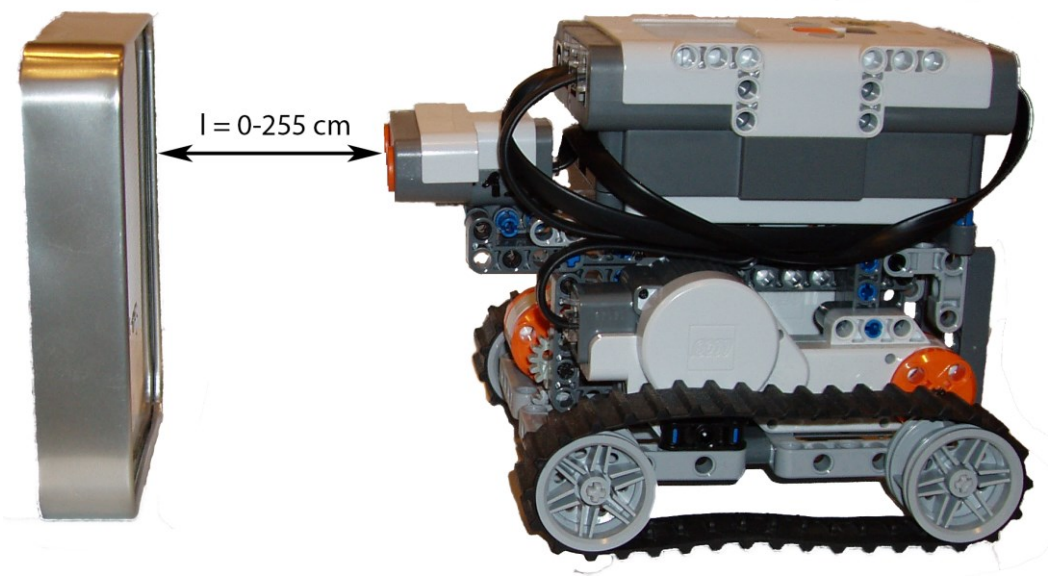


Obr. 4.11 Rychlosti motorů pro různé polohy trackballu

4.3.4 Ultrazvukový senzor

Pro měření vzdálenosti robotu od překážky byl použit ultrazvukový senzor, který byl umístěn v přední části robotu (Obr. 4.12). Pro čtení dat ze senzoru je v programovacím jazyce Bricx použita funkce *SensorUS (IN_1)*. Vstupním parametrem této funkce je port, ke kterému je senzor připojen. Výstupní hodnota vzdálenosti v rozsahu 0-225 je typu *byte*. V případě, že je vzdálenost *l* od překážky větší než 255*cm*, je senzorem odesílána hodnota 255.

Z důvodu, že řídicí jednotka nebyla schopna paralelně číst data ze senzoru a zároveň ovládat motory, je vzdálenost od překážky řídicí jednotce odesílána v časovém intervalu 3 sekund. Při odesílání vzdálenosti mobilnímu zařízení dojde na dobu 25*ms* k zastavení motorů. Pokud jsou tyto v chodu. Experimentálně bylo zjištěno, že v případě, kdy by časový interval odesílání vzdálenosti mobilnímu zařízení byl kratší, než 3 sekundy, nebyl by pohyb modelu plynulý a zastavení motorů by bylo postřehnutelné pouhým okem.



Obr. 4.12 Robot měřící vzdálenost od překážky ultrazvukovým senzorem

4.3.5 Ovládání hlasem

Od verze operačního systému Android 2.2 (*API* 8) je k dispozici aktivita pro rozpoznávání hlasu. Pro rozpoznávání hlasu pro systém s verzí Android 4.0.3 (*API* 15) a nižší musí být zajištěn přístup k internetu, protože samotné rozpoznávání probíhá na serverech společnosti Google. Pro verzi *API* 16 a vyšší již přístup k internetu není vyžadován a rozpoznávání hlasu probíhá offline. V době psaní této práce mi byl

k dispozici systém Android 4.0.3. Na (Obr. 4.13) je tak ukázka zdrojového kódu pro rozpoznávání hlasu online.

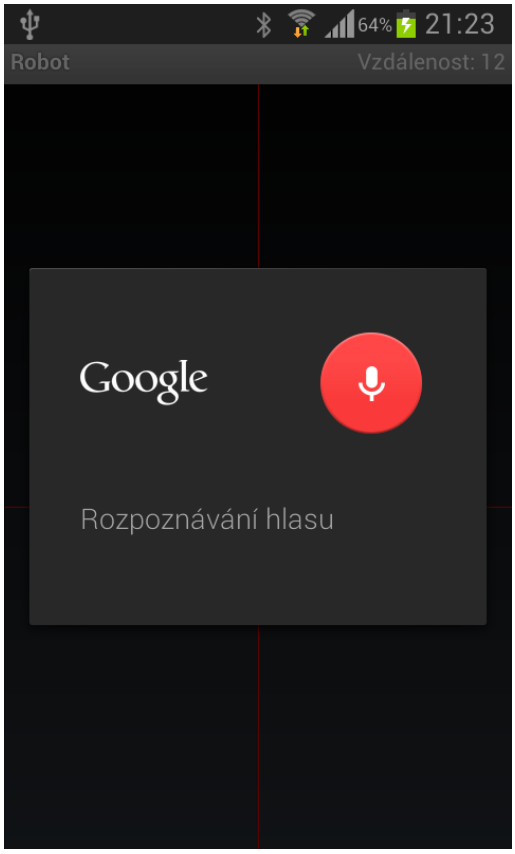
Funkce *StartVoiceRecognitionActivity* funguje následujícím způsobem. Třída *RecognizerIntent* spouští aktivitu *ACTION_RECOGNIZE_SPEECH*, která vyzve uživatele k řeči a vstup odešle k rozpoznání. Třída *RecognizerIntent* nastavuje model, který je preferovaný pro rozpoznávání řeči a výsledek je uložen do aktivity *onActivityResult*. Ve funkci *recognized* pak metoda *equals* porovnává řetězec *"dopředu"* se zadaným objektem. Pokud je řetězec s objektem shodný, metoda vrací hodnotu *true* a je vykonán příkaz pro pohyb robotu dopředu. V případě, že nenastala shoda, metoda vrací hodnotu *false*, příkaz není vykonán a aplikace se vrací do manuálního módu.

Pro ovládání modelu robotu hlasem byly předdefinovány příkazy *dopředu*, *dozadu*, *doleva* a *doprava*, na které robot reaguje. Jejich vykonávání není časově omezeno a lze je přerušit novým příkazem, nebo interakcí s displejem.

```
private void startVoiceRecognitionActivity()
{
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
    RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    intent.putExtra(RecognizerIntent.EXTRA_PROMPT, "Rozpoznávání hlasu");
    startActivityForResult(intent, VOICE_RECOGNITION_REQUEST_CODE);
}
private void recognized()
{
    String item;
    for(int i = 0; i < matches.size(); i++)
    {
        item = matches.get(i);
        if(item.equals("dopředu"))
        {
            for(int j = 0; j < 1; j++)
            {
                drv.LSpeed = 180;
                drv.RSpeed = 180;
            }
        }
        ...
    }
    ...
}
```

Obr. 4.13 Rozpoznávání hlasu

Po výběru možnosti hlasového ovládání v menu aplikace se uživateli zobrazí obrazovka s mikrofonem (Obr. 4.14), který při rozpoznávání hlasového vstupu tento stav signalizuje červeným podbarvením.



Obr. 4.14 Obrazovka aplikace při rozpoznávání hlasu

5 ZÁVĚR A ZHODNOCENÍ DOSAŽENÝCH VÝSLEDKŮ

Cílem práce bylo seznámit se s vývojem mobilních aplikací, popsat druhy komunikace mezi modelem a mobilním zařízením a ze stavebnice Lego Mindstorms vytvořit model, který bude ovládán pomocí mobilní aplikace. Z důvodu testování byla v jazyce C# vytvořena aplikace pro *PC*, která byla nejprve naprogramována pro příjem dat z NXT kostky. V prostředí Bricx byla následně vytvořena komunikace na straně NXT. Výsledkem této části bylo, že z NXT jednotky byl vysílán řetězec hodnot, který byl úspěšně přijat v programu vytvořeném v *PC*. Poté následovala časově nejnáročnější část, kdy byla v jazyce *C#* naprogramována funkce pro posílání textového řetězce do NXT zařízení, které jej vypsalo na displej. Takto byla ověřena obousměrná komunikace pomocí Bluetooth.

V další části této práce byl k NXT jednotce připojen servomotor a testován režim, ve kterém bude pracovat. Protože byla prioritou přesnost, byl vybrán režim, kde je motoru posílána rychlost, s jakou se má pootočit o jeden stupeň. Z dílů stavebnice Lego Mindstorms byl následně postaven model automobilu. U tohoto modelu se vyskytl problém s ovládáním přední nápravy, které bylo řešeno přes soustavu ozubených kol, což mělo za následek velkou vůli v řízení. Z tohoto důvodu byl motor umístěn tak, aby mohl být s řízením propojen napřímo. Ale i tato konstrukce vykazovala značnou vůli. Z tohoto důvodu byl postaven model robotu s diferenciálním podvozkem.

Poslední věc, která byla v jazyku *C#* testována, bylo ovládání robotu pomocí trackballu. Následně byl kód přepsán do jazyka Java a ovládání upraveno pro displej s rozlišením 480 x 800 *px*, pro který byla aplikace vytvořena. Pro vykreslení uživatelského rozhraní aplikace byla využita funkce, která vrací rozlišení displeje. Uživatelské rozhraní je tak vytvářeno dynamicky při startu aplikace. Třída *drive*, ve které je řešeno ovládání modelu robotu byla vytvořena pouze pro rozlišení 480 x 800 *px*, protože zařízení s jiným rozlišením displeje mi nebylo k dispozici a nemohla tak být testována správná funkčnost aplikace.

Model mobilního robotu lze ovládat interakcí s displejem, nebo hlasem. Pro verzi systému Android 4.0.3, pro kterou je aplikace zkompilována, probíhá rozpoznávání hlasového vstupu online. Pomocí grafického rozhraní aplikace je možné sledovat aktuální vzdálenost robotu od překážky.

Na straně NXT bylo nutné vyřešit paralelní chod obou motorů. K tomuto účelu byly využity funkce *Acquire* a *Release*. Experimentálně bylo zjištěno, že NXT jednotka není schopna pracovat s odezvou nižší než 25*ms*, což je pro účely této práce plně dostačující.

SEZNAM POUŽITÉ LITERATURY

BENNETT, James. Android Smartphone Activations Reached 331 Million in Q1'2012 Reveals New Device Tracking Database from Signals and Systems Telecom. *Android Smartphone Activations Reached 331 Million in Q1'2012 Reveals New Device Tracking Database from Signals and Systems Telecom*. 2012. Dostupné z: http://www.prweb.com/releases/2012/5/prweb9514037.htm

Eduxe [online]. 2013 [cit. 2013-05-01]. Dostupné z: http://www.eduxe.cz/product/9797-mindstorms-education-nxt-zakladni-souprava-332/

Gosphero [online]. 2013 [cit. 2013-05-01]. Dostupné z: http://www.gosphero.com/

HALL-GEISLER, Kristen. How Sphero Works. *How Sphero Works* [online]. 2012 [cit. 2013-03-06]. Dostupné z: http://electronics.howstuffworks.com/sphero1.htm

JAKEŠ, Tomáš. Řídící jednotka. *LEGO MINDSTORMS NXT - Robotické vzdělávání* [online]. 2013 [cit. 2013-03-04]. Dostupné z: https://www.lego.zcu.cz/web/ridici-jednotka

KOVAŘÍK, David. Operační systém v telefonu aneb nahlédněte do světa smartphonů. *Operační systém v telefonu aneb nahlédněte do světa smartphonů* [online]. 2012 [cit. 2012-07-04]. Dostupné z: http://mobilizujeme.cz/clanky/operacni-system-v-telefonu-aneb-nahlednete-do-sveta-smartphonu-vedecke-okenko/

LEGO Group. *NXT User Guide* [online]. 2006 [cit. 2013-03-06] Dostupné z: http://cache.lego.com/bigdownloads/buildinginstructions/4520736.pdf

NAGEL, Christian. *C# 2008: programujeme profesionálně*. Vyd. 1. Brno: Computer Press, 2009, 1126 s. ISBN 978-80-251-2401-7.

NEPŠINSKÝ, Roman. Google App Inventor – snadný vývoj aplikací pro Android. *Google App Inventor – snadný vývoj aplikací pro Android* [online]. 2010 [cit. 2012-07-04]. Dostupné z: http://www.svetandroida.cz/google-app-inventor-snadny-vyvoj-aplikaci-pro-android-201010

OPLETAL, Jiří. První smartphone na světě spatřil světlo světa přesně před 20 lety. *Svět Androida* [online]. 2012, č. 1 [cit. 2013-05-15]. Dostupné z: http://www.svetandroida.cz/ubehlo-20-let-od-uvedeni-prvniho-smartphonu-na-svete-201211

PAGE, Larry. Update from the CEO. In: *Google-Official Blog* [online]. 2013 [cit. 2013-05-13]. Dostupné z: http://googleblog.blogspot.co.uk/2013/03/update-from-ceo.html

Sphero Developer Center. *Www.gosphero.com* [online]. 2012 [cit. 2013-03-06]. Dostupné z: https://developer.gosphero.com/developers/get_started

SRB, Jindřich. Samsung Galaxy S II, Galaxy Tab 10.1 a Wave 578. *Mobilizujeme.cz* [online]. 2011, 1. [cit. 2013-05-15]. Dostupné z: http://mobilizujeme.cz/clanky/samsung-galaxy-s-ii-galaxy-tab-10-1-a-wave-578/

TIŠNOVSKÝ, Pavel. Eclipse – integrované vývojové prostředí pro Javu i další programovací jazyky. *Eclipse – integrované vývojové prostředí pro Javu i další programovací jazyky* [online]. 2012 [cit. 2012-07-04]. Dostupné z: http://fedora.cz/eclipse-integrovane-vyvojove-prostredi-pro-javu-i-dalsi-programovaci-jazyky/

TOKÁR, Daniel. Nativní vs. webový vývoj aplikací na mobilní platformy [online]. 2011 [cit. 2012-02-21]. Bakalářská práce. Masarykova univerzita, Fakulta informatiky. Vedoucí práce Jaroslav Škrabálek. Dostupné z: http://is.muni.cz/th/325070/fi_b/

ZAKHOUR, Sharon. *Java 6: výukový kurz*. Vyd. 1. Brno: Computer Press, 2007, 534 s. ISBN 978-80-251-1575-6.